Constructs

The three control constructs are

- `if`
- `do`
- `case`

The `if` construct is just as it has been since Fortran 77, except that it may use construct names.

Learn more about the `if` construct.

Next slide

Blocks

A *block* is a collection of statements that are executed all together under control of an `if`, a `do`, or a `case` statement.

Previous slide Next slide

Construct Names

Any of the constructs may have a construct name, followed by a colon on the first line of the construct. The `end if`, `end do`, or `end select` statement that ends the construct must be followed by the same construct name.

Previous slide Next slide

The case Construct

```
traffic_light: select case (light_color)
   case ("red")
      call stop_the_car
      stop
   case ("yellow")
      call go_slowly
   case ("green")
      call step_on_the_gas
end select traffic_light
```

Learn more about the `case` construct.

Previous slide Next slide

There may be several values in a `case` statement, separated by commas.

There may be a range of values, indicated by a colon.

The values in the case statement must all be determined at compile time (constants).

Values in different case statements must not overlap.

Learn more

Previous slide Next slide

```
function number_of_days (month, year)
   implicit none
   integer month, year, number_of_days
   logical leap_year
   external leap_year

   select case (month)
      case (4, 6, 9, 11)
         number_of_days = 30
      case (1, 3, 5, 7:8, 10, 12)  ! Range of values
         number_of_days = 31
      case (2)
         if (leap_year (year)) then
            number_of_days = 29
         else
            number_of_days = 28
         end if
      case default
         number_of_days = 0
   end select
end function number_of_days
```

Previous slide Next slide

Exercise

1. Write a program using a `case` construct that prints the word ``vowel" if the value of the variable `letter`, read from an input file, is a vowel (i.e., a, e, i, o, or u), prints the word ``consonant" if the value of `letter` is any other letter of the alphabet, and prints ``not a letter" if it is any other character.

Previous slide Next slide

do Construct

All forms of the `do` construct except those that end with `end do` should be avoided.

There are three sorts of `do` constructs: `do`, `do` with iteration count, and `do while`. The `exit` and `cycle` statements are used to exit the loop and begin the next iteration of the loop. The first example has an iteration count and an `exit` statement.

Learn more about the `do` construct.

Previous slide Next slide

```
search_loop: do i = 1, table_size
   if (item == table (i)) then
      location = i
      exit search_loop
   end if
end do search_loop
```

Previous slide Next slide

```
do
   read (*, *, iostat = ios) data
   if (ios > 0) cycle    ! Error
   if (ios < 0) exit     ! End of file
   call process (data)
end do
```

Previous slide Next slide

```
converged = .false.
do while (.not. converged)
   call iter_8 (data, converged)
end do
```

Learn more

Previous slide Next slide

Case Study: Approximating a Definite Integral

The value of a definite integral is the area of a region of the plane bounded by the three straight lines. $x = a$, $y = 0$, $x = b$, and the curve $y = f(x)$.

If the curve $y = f(x)$ is replaced by a straight line with endpoints a and b, the region in question is converted to a trapezoid, a simple four-sided figure whose area is given by the formula

$A = (b - a) \times [f(a) + f(b)] / 2$

Previous slide Next slide

The area under the curve $y = \sin(x)$, for x from 0 to *pi* radians (180°) is the integral from 0 to *pi* of $\sin(x)$

If the width of each trapezoid is h, we have the relationship

$h = (b - a) / n$

The sum of the areas of the n trapezoids may be expressed by the formula

$T_n = h [f(a) / 2 + f(a + h) + f(a+2h) + ... + f(b-h) + f(b) / 2]$

Previous slide Next slide

In the following program `integral`, the sum is formed by first computing

$f(a)/2 + f(b)/2 = 0.5 \times [f(a) + f(b)]$

The integer variable `i` counts 1, 2, ..., n-1 and computes the expression `a + i * h` to obtain the sequence of values

$a+h, a+2h, ..., a+(n-1)h = b-h$

Previous slide Next slide

```
program integral
!  Calculates a trapezoidal approximation
!  to an area using n trapezoids.
!  n is read from the input file.

!  The region is bounded by lines x = a,
!  y = 0, x = b,  and the curve y = sin (x).
!  a and b also are read from the input file.

   implicit none
   real :: a, b, h, sum
   integer :: i, n
```

Previous slide Next slide

```
read *, n
print *, "Input data  n:", n
read *, a, b
print *, "Input data  a:", a
print *, "            b:", b
```

Previous slide Next slide

```
   h = (b - a) / n

!  Calculate the sum f(a)/2 + f(a+h) +...
!                    +f(b-h) + f(b)/2
!  Do the first and last terms first
   sum = 0.5 * (sin (a) + sin (b))
   do i = 1, n - 1
      sum = sum + sin (a + i * h)
   end do

   print *,  &
   "Trapezoidal approximation to the area =", &
   h * sum
end program integral
```

Previous slide Next slide

Exercises

1. An integer is a perfect square if it is the square of another integer. For example, 25 is a perfect square because it is 5x5. Write a program to selectively print those numbers less than 100 that are *not* perfect squares.
2. Using the fact the `selected_real_kind` and `selected_int_kind` return a negative value when asked to produce a kind number for a precision or range not available on the system, determine all the possible kind numbers for reals and integers on your system.

Previous slide